View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Archivio della ricerca - Università degli studi di Napoli Federico II

Generating and Instantiating Abstract Workflows with QoS User Requirements

Claudia Di Napoli[1], Luca Sabatucci[2], Massimo Cossentino[2] and Silvia Rossi[3]
[1]*C.N.R., Istituto di Calcolo e Reti ad Alte Prestazioni, Napoli, Italy*
[2]*C.N.R., Istituto di Calcolo e Reti ad Alte Prestazioni, Palermo, Italy*
[3]*DIETI, Università degli Studi di Napoli Federico II, Napoli, Italy*

Keywords: Service-oriented Computing, Automatic Service Composition, Multi-agent Negotiation.

Abstract: The growing availability of services accessible through the network makes it possible to build complex applications resulting from their composition that are usually characterized also by non-functional properties, known as Quality of Service (QoS). To exploit the full potential of service technology, automatic QoS-based composition of services is crucial. In this work a framework for automatic service composition is presented that relies on planning and service negotiation techniques for addressing both functional and non-functional requirements. The proposed approach allows for dynamic service composition and QoS attributes, and it can be applied when services are provided in the contest of a competitive market of service providers without knowledge disclosure.

1 INTRODUCTION

Service-oriented computing (Papazoglou et al., 2007) is opening new frontiers in the development of added value applications, known as Service-Based Applications (SBAs), resulting from the combination of different services provided by different providers at different conditions, so leading to the vision of an open market of services regulated by demand and supply mechanisms.

In this context, the value of commercial Service-Based Applications (SBAs) depends not only on the functionality they deliver, but also on the value of their non-functional properties, known as Quality of Service (QoS), that are not tied to the specific functionality, but rather to "how" it is provided, i.e. how well a service serves the customer (Strunk, 2010), in terms of its non-functional aspects (e.g. performance, reliability, availability, cost). In fact, service implementations having similar functional capabilities are distinguishable for their QoS values that determine whether a service is reliable, trustworthy, or efficient, since it may be functionally capable of performing a given task, but it might not be reliable or efficient enough in performing the task up to the user satisfaction (Shehu et al., 2014).

When dealing with applications composed of services orchestrated according to a specified workflow, QoS values are usually specified at the application level, rather than at level of the single components, and they result from the aggregation of the QoS values of each component service, according to a function depending on the nature of the considered QoS parameters, and on the structure of the SBA workflow These end-to-end QoS values depend on the values of each component service, aggregated according to a function usually depending on the nature of the considered QoS parameters, and on the structure of the SBA workflow.

In order to automatically compose QoS-aware SBAs in response to a user request with QoS constraints, several approaches have been proposed in the literature (Moghaddam and Davis, 2014). Static approaches typically assume that QoS parameters are pre-defined for each service, and they do not change during the composition process. These approaches are not suitable within the vision of a dynamic market of services, since QoS values may vary according to the provision strategies of providers as well as to users' requirements expressed as global constraints on the SBA's QoS. So, heuristic approaches leading to sub-optimal solutions are often adopted where service QoS values are not known prior to generating the workflow, reflecting a more realistic scenario.

In the present work, we present an integrated framework that allows performing all stages of an

Di Napoli C., Sabatucci L., Cossentino M. and Rossi S.
Generating and Instantiating Abstract Workflows with QoS User Requirements.
DOI: 10.5220/0006203902760283
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 276-283
ISBN: 978-989-758-219-6
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

automatic QoS-based service composition process, ranging from finding functionally equivalent compositions, to the selection of an appropriate composition that meets the non-functional user's constraints. The framework allows to deal with cases in which i) there could be more plans (i.e. combinations of services providing a complex service) to satisfy the same set of user requirements, and ii) there could be more providers for the same service in a plan. In fact, a complex service can be realized by different plans, and each plan may be composed of a high number of tasks. In a real scenario, each task may be possibly accomplished by different services providing the same functionality. When there is a high number of possible service combinations providing the complex task, the problem of selecting an optimal combination that meets the QoS values specified by a request becomes intractable. The framework provides an automatic reasoner, and it is equipped with automated negotiation capabilities. The reasoner produces several workflows of services (or plans) by aggregating different combinations of capabilities. Each workflow represents an alternative solution for addressing the specified functional part of a user request. The automated negotiation –upon QoS attributes values of single services– allows to select services whose QoS values, once aggregated, meet the QoS constraints set by the user.

The paper is organized as follows: Section 2 reports some related work on QoS based service selection in service composition. Section 3 focuses on the integration of the approach for service composition with the strategy for service selection. An example of a QoS based complex service composition in the domain of Travel Service is presented in Section 4. Finally, some conclusions are drawn in Section 5.

2 QoS BASED SERVICE SELECTION

In QoS-aware service composition, the selection of service implementations is based on the QoS values of the candidate services. When the QoS values are expressed at the level of the complete application, the problem of finding a combination of services whose QoS values, once aggregated, meet the QoS requirements is a constrained optimization problem that can be reduced to the Multi-Choice Knapsack problem known to be NP-hard. Several approaches propose algorithms to optimize the global QoS values of a service composition subject to multiple QoS constraints (Zeng et al., 2004), (Ardagna and Pernici, 2007), (Yu et al., 2007), (Alrifai and Risse, 2009). These approaches rely on prior knowledge of the component service QoS values. They do not consider dynamic changes in QoS, so they are not suitable in an open market of services where QoS values may constantly change over time, as it is assumed in the present work.

Heuristics approaches have been investigated to find near-optimal combinations in dynamic environments. In this research line, negotiation mechanisms were shown to be a suitable approach to deal with QoS-aware SBAs (Yan et al., 2007), (Di Napoli et al., 2014), (Lau, 2007).

Negotiation solutions are divided by the assumption that the service providers are or not predetermined before negotiation. The two approaches are known as pre-contractual negotiation, and dynamic provider selection. Pre-contractual negotiation usually applies negotiation for each required service independently from the others, relying on bilateral one-to-one negotiation mechanisms such as in (Yan et al., 2007), (Paurobally et al., 2007), (Siala and Ghedira, 2011), not allowing competition among providers. In other approaches negotiation occurs after an optimization phase, but only among the providers that do not provide the expected QoS local values (Ardagna and Pernici, 2007). The approach in (Di Napoli et al., 2014) proposes a coordinated negotiation mechanism with all providers of the different services in the composition, taking into account their contribution to the global QoS values. The negotiation occurs concurrently with all service providers, and it requires a coordination point to synchronize the concurrent negotiations for global evaluation. This approach allows to exploit the competition among service providers without excluding less promising providers that could become more competitive while negotiation proceeds.

3 AUTOMATIC QoS-AWARE SERVICE COMPOSITION

The framework proposed for automatic composition of services with specified QoS user's preferences allows to address both functional and non-functional users requirements by integrating *MUSA*, a middleware for service composition and orchestration (Sabatucci and Cossentino, 2015; Sabatucci et al., 2016), with a QoS based service negotiation module (Di Napoli et al., 2014), hereafter *QoS Negotiation*.

The overall process for QoS-aware service composition is split in five phases:

1. the formulation of a user's goal specifying a required functionality, together with non-functional requirements expressed in terms of QoS values,

2. automatic composition of abstract plans to address the goal via AI planning techniques,
3. service discovery, i.e. the identification of concrete services matching the functional requirements of the tasks that compose the abstract plan,
4. QoS service selection, i.e. the identification of one concrete service implementation for each task of the plan whose QoS values, once aggregated with the values provided by the other services in the composition, satisfy the user's preferences,
5. service binding, i.e. instantiation of the abstract plan into an executable workflow, ready to be enacted.

Steps 1 and 2 are conducted by MUSA, whereas the QoS Negotiation enacts steps 3, 4 and 5.

3.1 MUSA

MUSA (Middleware for User-driven Service Adaptation) (Sabatucci et al., 2016; Sabatucci et al., 2015) is a middlware for composing and orchestrating distributed services according to unanticipated and dynamic user needs. It is a platform in which 1) virtual enterprises can deploy capabilities that wrap real services, completing them with a semantic layer for their automatic composition; 2) analysts and/or users can inject their goals for requesting a specific outcome. Under the hypothesis that both goals and capabilities refer to the same ontology, agents of the system are able of composing available services into plans for addressing the user request.

The main abstraction in MUSA is a *capability*, which represents something that the system knows how to do, and *goals*, used to represent the user's requirements. In MUSA, a user is able to describe a desired functionality via a high-level goal-oriented description language, called GoalSPEC (Sabatucci et al., 2013). At run-time, once a goal model is specified (often it is a tree hierarchy of goals), it may be injected in the system. This event triggers the formation of an agent group that is committed to address the goal (and all its subgoals). Goals represent conditions, in terms of states of the world, a user may want to be addressed.

A proactive means-end reasoning (Sabatucci and Cossentino, 2015) associates system capabilities to user goals for deducing possible service compositions that can guarantee the desired final state. This procedure works on two main abstractions: the *Abstract Capability* and the *Goal*.

MUSA also provides a capability language, based on predicate logic, that allows to specify additional information about a service (Sabatucci et al., 2016; Sabatucci et al., 2015). The description of a service is made of two parts: an abstract part and a concrete part. The Abstract Capability represents a generalization of a service, and it forces the idea of categories of services that produce similar results but that are offered by different providers. It offers a descriptor for the common elements of a category of similar services: pre and post conditions, to be checked before and after service execution, and evolution, to be used in order to simulate the effects of the service for enabling the automatic composition.

The Configurator group is an emerging organization of agents whose responsibility is to aggregate capabilities in order to address a given goal through a procedure called *Proactive Means-End Reasoning* (Sabatucci and Cossentino, 2015). This procedure works on two main abstractions: the *Abstract Capability* and the *Goal*. Goals represent conditions, in terms of states of the world a user may want to be addressed.

When more Concrete Capabilities are available for the same Abstract Capability, multiple Concrete Workflows could be obtained, so a selection among them has to be made. To date, MUSA does not prescribe a specific strategy for service selection among concrete services that may provide different QoS.

3.2 QoS Negotiation

In the approach proposed in (Di Napoli et al., 2014), negotiation is used as a dynamic service selection mechanism, aiming at selecting the services that best match the service requester's non-functional requirements, allowing to manage the dynamic nature of QoS values. In fact, as pointed out, while a service functionality is static, and it is a characterizing intrinsic feature of the service itself, its non-functional features may vary, as well as users' QoS requirements/preferences. This is even truer when services are provided in an open market of services, as it happens already for several classes of services in different application domains where services are considered digital goods to be purchased.

The negotiation occurs between the service requester, represented by a software agent, we refer to as the *Service Negotiator* (SN), and the candidate service providers, represented by software agents, we refer to as the *Service Provider*s (SPs). They negotiate upon the QoS values of the single services whose functionalities are required in the composition, reaching an agreement if the local QoS values of each service, once aggregated, meet the global QoS requirements specified by the user when the goal specification is injected in the system. So, a successful

negotiation selects the composition of services that best matches the service requester's non-functional requirements.

The adopted negotiation mechanism is based on an iterative protocol, where the negotiator is the *initiator* acting on behalf of the service requester, and the providers formulate offers specifying the QoS values of the service they are able to provide (Di Napoli et al., 2014). The Service Negotiator evaluates the received offers, without issuing counteroffers, since, as it happens in a real market of services, it is unlikely to have enough information on the providers' strategies to formulate counteroffers. In addition, in a service composition, counteroffers for a single functionality cannot be formulated independently from the ones received for the other functionalities. In other words, negotiating over the attributes of the single abstract services cannot be done independently from each other. Single counteroffers could be formulated only if negotiation occurs on one service after another, since in a composition of services changing the QoS values of one service can be done only assuming that the QoS values of the other services are fixed. This approach would result in serializing the negotiation, so leading to a long process depending on the number of abstract services in a workflow, not suitable in a dynamic market of services (Di Napoli et al., 2015).

The negotiation mechanism allows the negotiator to negotiate with all providers of services with the same required functionality required in the abstract workflow, and to evaluate if the aggregated QoS values of the received offers meet the user QoS requirement, so to decide whether or not to accept the offers.

The negotiation protocol is organized in a set number of negotiation *round*s, until a *deadline* is reached, or the negotiation is successful. The deadline is the number of allowed rounds. Within a round, the negotiator concurrently negotiates with all available providers both with the ones having the same functionality, and with the ones having different functionalities. The end of a negotiation round represents a synchronization point allowing the negotiator to evaluate the global QoS values, that is a necessary coordination step to negotiate for end-to-end QoS values.

The negotiation process starts when an *Abstract Workflow* (AW) is produced. It is a directed acyclic graph $AW = (AS, P)$ where $AS = AS_1, \ldots AS_n$ is a set of nodes, and P is a set of directed arcs. Each node represents an *Abstract Service* (*AS*), i.e. a generic service description that specifies a required functionality in the composition of services. The Service Negotiator prepares m call for proposals (*cfp*s), one for each Abstract Service in the Abstract Workflow (AW), each one sent to the set of n available providers for that Abstract Service. So, at each round $m * n$ *cfp*s are sent. After waiting for the time set to receive offers (i.e., the *expiration time* of a negotiation round), the negotiator checks if there are offers for each Abstract Service; if not, it declares a failure since it is not possible to find a Concrete Service corresponding to each Abstract Service. Otherwise, it evaluates a selected set of offers, one for each Abstract Service, and according to the result of the evaluation, it performs one of the following actions:

- if the aggregated QoS values of the selected offers do not meet the user QoS request, it asks for new offers by sending again $m * n$ *cfp*s, so starting another negotiation round with all the providers;
- if the aggregated QoS values of the selected offers meet the user QoS request, it accepts the offers sent by the corresponding providers (one for each abstract service), so ending the negotiation successfully;
- if the deadline is reached without a success, the negotiator declares a failure to all providers that took part in the negotiation.

In order to check if there is a combination of offers that satisfies the end-to-end QoS constraint, at the end of a negotiation round t the Service Negotiator selects the most promising offer for each Abstract Service, and then it verifies if the aggregation of the selected offers meet the user's constraints, to avoid the computation of all possible combinations of offers received for each Abstract Service. In the case of additive QoS parameters, and QoS user requirements representing an upper bound for acceptable compositions, a promising offer $\bar{x}_{i,j}$ at round t for AS_i is computed according to the approach proposed in (Alrifai and Risse, 2009), as follows:

$$\bar{x}^t_{i,j} = \underset{x^t_{i,j} \in AS_i}{argmax}\left(\frac{max(x^t_{i,j}) - x^t_{i,j}}{\sum_{k=1}^{m} max(x^t_{k,j}) - \sum_{k=1}^{m} min(x^t_{k,j})}\right) \quad (1)$$

where m is the number of Abstract Services composing the Abstract Workflow, $max(x^t_{i,j})$ is the maximum $x^t_{i,j}$ value offered by the service provider j for the QoS parameter among all the available offers for the AS_i at time t, while $min(x^t_{i,j})$ is the corresponding minimum $x^t_{i,j}$ attribute value. Equation 1 estimates how promising an offer is w.r.t. the other offers for the same Abstract Service (local evaluation), and to the entire workflow (global evaluation) by providing an indication of how good the value of each QoS parameter is w.r.t the QoS values offered by other providers of the same Abstract Service (the numerator), in relation to the possible aggregated values of the same

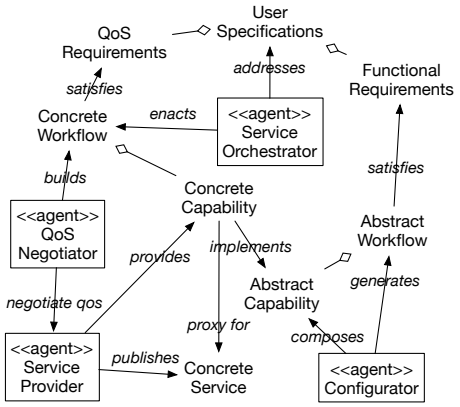

Figure 1: Conceptual integration of MUSA and QoS Negotiation components.

parameter for all the Abstract Services (the denominator).

Then, the Service Negotiator checks if the selected set of offers at round t meets the constraint, i.e.:

$$\sum_{i=1}^{m} \vec{x}^{t}_{i,j} \leq Req\_QoS \tag{2}$$

where Req_QoS is the user required QoS value, expressed as an upper bound of an acceptable combination, and it is assumed to be composed of one QoS parameter.

3.3 System Integration

Figure 1 shows an overview of the integration between the architecture of MUSA and the components for the automatic negotiation. The integration was possible thanks to a semantic reconciliation:

- MUSA introduces the concept of Abstract Capability in order to provide a means for describing a service in abstract terms. The QoS Negotiation uses Abstract Service (AS) for the same concept.
- A MUSA plan –built for addressing a set of goals– corresponds to the QoS Negotiation Abstract Workflow (AW).
- In MUSA, each Abstract Capability is associated to a Concrete Capability that encapsulates the instructions for invoking a specific service. Given that more providers may offer the same functionality, in order to distinguish them the QoS Negotiation refers to a specific Concrete Service (CS).
- Finally, the run-time plan, that will be executed by the workflow engine, corresponds to a Concrete Workflow (CW).

QoS metrics to compute end-to-end QoS values and aggregation rules, depend both on the types of QoS, and on the structure of the AW (i.e., its control flow). In the integrated framework, service selection, as shown in Figure 2, is carried out before execution since starting the execution of a Concrete Workflow that can end up without meeting the users' QoS requirements would result in a waste of computing resources.

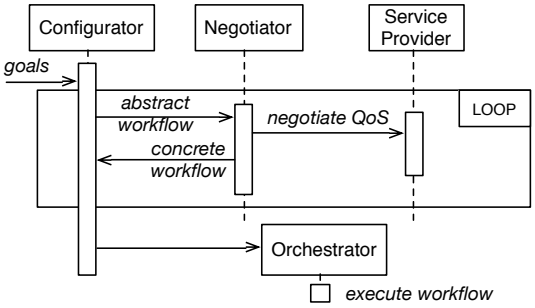

Figure 2: Interaction diagram showing the integration of MUSA and QoS Negotiation components.

Once one or more plans or Abstract Workflows have been generated, a set of concrete capabilities are mapped to each abstract capability in the workflow. Concrete capabilities are implemented by Web Services, and as such they only know about themselves, and they are not capable of autonomous action, intentional communication, or proactive cooperative behavior (Huhns, 2002). So, in order to provide the middleware with negotiation capabilities, a software agent approach is adopted as proposed in (Di Napoli, 2009). In fact, software agents possess all of these meta-capabilities, and they are used to represent service providers and service consumers.

4 A MOTIVATING EXAMPLE: THE SMART TRAVEL SERVICE

A Smart Travel service is a complex distributed composite service acting as a tour operator for organizing holiday packages and supporting travelers on-the-run. The user may use the Smart Travel service to organize a vacation by specifying a set of functional requirements about the kind of desired vacation including the geographic area of interest, places of interest, activities to perform, and a ser of non functional requirements about a budget to spend, or the degree of service reliability and so on.

Users may express travel preferences supported by a flexible language and a specific interface to convey her goals about: places to visit, activities to do and –in general– the kind of vacation. The aim is to aggregate heterogeneous touristic services (flights, trans-

fers, hotels and tickets for museums, for the opera and for other local events) in a dynamic, open and geographically distributed environment. Creating new travel experiences grounds on putting traveler's requirements at the center of the process.

The system has to organize the corresponding services and acts as a local guide for traveler, running on a personal device.

4.1 Abstract Workflow Generation

In the running example goals represent the main instrument for the users in order to express their traveling requests. An example could be the goal:
WHEN in sicily THE user SHOULD ADDRESS visiting palermo for at least 3 days.

Besides the functional requirements, quality assets are the fundamental mechanism to increase user's flexibility in expressing their expectations. They represent a powerful instrument to set the expected QoS. An example of QoS asset is:
the total cost IS LOWER THAN 1500 €.

An example of a goal representing a travel request is the following:

```
GOAL to_visit_city:
  WHEN number_nights_to_spend(city,nights)
       AND nights > 0
  THE SYSTEM SHALL ADDRESS
  hotel_reserved(city)
```

For instance, HotelReservation is the abstract capability that identifies the reservation of a generic *hotel* in a city for a number of nights. It is described as follows:

```
capability(hotel_reservation,

  % semantic description
  precondition(available(city,dates)),
  postcondition(hotel_reserved(city)),
  evolution(add(hotel_reserved(city))),

).
```

Clearly, many Concrete Capabilities may refer to the same abstract capability by offering the service with different QoS (for example, prices, stars, ...). Here we show the Concrete Capability provided by Expedia services. The Concrete Capability also specifies technical details for invoking the web-service front-end.

```
concrete_capability(expedia_hotel_reservation,
  extends(hotel_reservation)

  % service interface
  method(https,put),
  address(http://terminal2.expedia.com/x/
       mhotels/reserve),
  input([
    param(city,string),
    param(room,integer),
    param(checkInDate,date),
    param(checkOutDate,date)]),
  output([param(reservation_id,string)]),

  % protocols
    service_protocol(rest),
    require_auth(oauth20)
).
```

According to the available services, agents of the system are able of producing more plans for a required goal. The proactive means-end reasoning works with abstract capabilities, and it adopts a symbolic approach to satisfy the goal from a functional point of view. Therefore the resulting workflow aggregates generic categories of services.

In the context of the Smart Travel service, the proactive means-end reasoning organizes a trip by combing a set of traveling acts: visiting a touristic place, attending a local event, relaxing on a beach/resort/natural place, and moving from a place to another one. An example of trip is: 1. flight to dest_city, 2. visit for 2 days, 3. move to beach_resort, 4. relax 1 day, ... 13. flight back.

As a consequence the algorithm deduces an abstract workflow for arranging the services necessary for the trip. For instance, flying to a city demands to book a seat in a flight in a given data, from the departure city to the destination (flight_booking capability); visiting a city requires to reserve an accommodation for the corresponding nights of visit (hotel_reservation capability); and so on. The output is a workflow that aggregates some abstract capabilities. For example, an extract of the workflow is the following sequence: 1. flight_booking, 2. hotel_reservation, 3. restaurant_booking.

The plan is still abstract because it requires additional data, for instance about the kind of accommodations, the timetables, and the price. Clearly, many alternatives exist for each single service, thus potentially leading to a number of combinations.

In order to become operative, it is necessary to specify, for each category of services, the concrete service (and therefore its provider) that will be invoked. The specified quality assets help the system to select concrete services. For instance, the requirement about the budget helps the system to discard luxury hotels providers from the list to be contacted. To this aim, QoS-based service selection plays a central role. Indeed, in order to address QoS-aware service composition, it is necessary to select the combination of services that satisfies also the non-functional requirements, expressed as end-to-end QoS values on the complete plan.

4.2 Service Providers Strategy

The adopted negotiation mechanism is unilateral, i.e. counteroffers are not proposed by the negotiator. So, in order for the negotiation to converge, the only strategy the providers can adopt is a concession

one. Several types of concession strategies in automated agent negotiation are proposed in the literature (Faratin et al., 1998), (Lopes and Coelho, 2010). Here, the concession strategies are modeled as multi-dimensional Gaussian functions, where each dimension represents a single QoS attribute to be negotiated (Rossi et al., 2016). The Gaussian function represents the provider's utility of the generated offers, and at the same time the probability distribution of the offers the provider may generate. The use of Gaussian functions allows to simulate a stochastic behaviour of service providers with zero-intelligence, by approximating the trends of a volatile and open market of services. A Gaussian function G is characterized by its parameters μ and σ ($G(\mu,\sigma)$). The best offer in terms of the provider's own utility is represented by the μ value of the Gaussian function, i.e. $U(\mu) = 1$, and it corresponds to the QoS value of the offer with the highest probability to be generated. The Gaussian standard deviation σ represents the attitude of the provider to concede during negotiation, and it determines its offer reservation value ($\mu - \sigma$). Hence, bigger values of σ correspond to smaller reservation values, so to more conceding providers. The negotiation set of a QoS parameter is $[\mu - \sigma; \mu]$.

At each negotiation round, each provider generates, following its probability distribution, a new utility value corresponding to a new offer. If this value is lower than the one offered at the previous round and within the negotiation set, then it proposes the new value. If this value is greater than the one offered at the previous round, or it is outside the negotiation set, the provider proposes the same value offered in the previous round. This strategy allows to simulate different and plausible behaviours of providers that prefers not having a consistent loss in utility, even though when the number of negotiation rounds increases, the probability for the provider to move towards its reservation value increases.

4.3 Negotiating upon Smart Travel QoS

Once MUSA processes the injected user goal for the Smart Travel example, an Abstract Workflow composed of 3 Abstract Services is computed: flight service, hotel service, entertainment services. For each service more providers are available and the negotiation among them takes place in order to select the combination of Concrete Services that meets the user's requirement on the cost parameters. At each round providers submit their offers, and at the end of the round the negotiator selects a set of promising offers, one for each Abstract Service, and it checks if the corresponding cost is less than the required global values, i.e., $\sum_{i=1}^{m} \bar{x}_{i,j} \leq 1500€$.

If no solution is found at round t, negotiation takes place with all providers at round $t+1$. In fact, since the strategies of negotiators are not known to the negotiator for the knowledge disclosure assumption, there is no need to discharge providers that could become more competitive during the negotiation process by updating their offered QoS values. If negotiation is successful, a sub-optimal solution is found w.r.t. the user request. Of course, the length of negotiation in terms of the negotiation rounds necessary to reach an agreement, if any, depends on both the providers' strategies, and on their attitude to concede, but also on the distribution of the QoS values among the component services. The simulation of negotiation upon the QoS price value of the Smart Travel service is carried out by randomly generating one Gaussian distribution for each provider, with different concession rates, and reservation values. It should be noted that providers of the same service have the same optimal QoS value for their offer (i.e., the one with utility value equal to 1), to simulate that the desired price of a service is an agreed value among the providers to avoid an unbalanced market. According to their Gaussian distribution, providers generate their offers at each negotiation round, until they are informed either of the negotiation success, or of the negotiation end with a failure. The negotiation success is reached when the compositor finds a composition of offers with any QoS value solution of Equation 1. Concurrent negotiation with all providers at each round makes the negotiation time independent from the number of Abstract Services in the composition, and of the providers available for each of them.

5 CONCLUSIONS

The paper proposes a QoS based service composition framework that relies on both AI planning, and automatic agent negotiation: the first allows to generate abstract workflows satisfying user's goals, while the second allows to select the service providers able to satisfy the QoS requirements specified by the user. This hybrid approach allows to address the challenges due to the dynamic nature of an open system providing services that are created and updated on the fly. In fact, such a system has the same characteristics of a market driven by the supply and demands of goods whose non-functional characteristics have values that cannot be fixed a priori. In fact, they depend on several factors as the volatility of the market that may affect providers' strategies, the nature of the provided services, and the users' preferences. In addi-

tion, the proposed approach allows concurrent negotiations among providers of the same and the different services required in the composition, making the negotiation time independent from the number of both abstract services in the composition, and of service providers for each abstract service, that is a crucial requirement in service oriented scenarios.

ACKNOWLEDGEMENTS

This work has been partially supported by the Italian Ministry of Education University and Research within the PRIN 2015 Project "Profiling and Adaptation for User-centered Assistive Robotics" (Ref. 2015KBL78T).

REFERENCES

Alrifai, M. and Risse, T. (2009). Combining global optimization with local selection for efficient QoS-aware service composition. In *Proceedings of the 18th Int. Conf. on World Wide Web*, pages 881–890. ACM.

Ardagna, D. and Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Trans. on Software Eng.*, 33(6):369–384.

Di Napoli, C. (2009). *Knowledge Processing and Decision Making in Agent-Based Systems*, chapter Software Agents to Enable Service Composition through Negotiation, pages 275–296. Springer Berlin.

Di Napoli, C., Di Nocera, D., Pisa, P., and Rossi, S. (2014). A market-based coordinated negotiation for qos-aware service selection. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, volume 187 of *LNBIP*, pages 26–40. Springer.

Di Napoli, C., Di Nocera, D., and Rossi, S. (2015). Computing pareto optimal agreements in multi-issue negotiation for service composition. In *Proceedings of AAMAS 2015*, pages 1779–1780.

Faratin, P., Sierra, C., and Jennings, N. R. (1998). Negotiation Decision Functions for Autonomous Agents. *Robotics and Autonomous Systems*, 24:3–4.

Huhns, M. N. (2002). Agents as web services. *IEEE Internet Computing*, 6(4):93–95.

Lau, R. Y. K. (2007). Towards a web services and intelligent agents-based negotiation system for B2B ecommerce. *Electronic Commerce Research and Applications*, 6(3):260–273.

Lopes, F. and Coelho, H. (2010). *Concession Behaviour in Automated Negotiation*, pages 184–194. Springer Berlin Heidelberg.

Moghaddam, M. and Davis, J. (2014). *Service Selection in Web Service Composition: A Comparative Review of Existing Approaches*, pages 321–346. Springer New York, New York, NY.

Papazoglou, M., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45.

Paurobally, S., Tamma, V., and Wooldrdige, M. (2007). A framework for web service negotiation. *ACM Trans. Auton. Adapt. Syst.*, 2(4).

Rossi, S., Di Nocera, D., and Di Napoli, C. (2016). *Recent Advances in Agent-based Complex Automated Negotiation*, volume 638 of *Studies in Computational Intelligence*, chapter Gaussian-Based Bidding Strategies for Service Composition Simulations, pages 193–208. Springer International Publishing.

Sabatucci, L. and Cossentino, M. (2015). From Means-End Analysis to Proactive Means-End Reasoning. In *Proceedings of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Florence, Italy*.

Sabatucci, L., Lodato, C., Lopes, S., and Cossentino, M. (2015). Highly customizable service composition and orchestration. In *Service Oriented and Cloud Computing*, volume 9306 of *LNCS*, pages 156–170. Springer.

Sabatucci, L., Lopes, S., and Cossentino, M. (2016). A goal-oriented approach for self-configuring mashup of cloud applications. In *Cloud and Autonomic Computing (ICCAC), 2015 International Conference on*.

Sabatucci, L., Ribino, P., Lodato, C., Lopes, S., and Cossentino, M. (2013). Goalspec: A goal specification language supporting adaptivity and evolution. In *Engineering Multi-Agent Systems*, pages 235–254. Springer.

Shehu, U., Epiphaniou, G., and Safdar, G. A. (2014). A survey of qos-aware web service composition techniques. *International Journal of Computer Applications*, 89(12):10–17.

Siala, F. and Ghedira, K. (2011). A multi-agent selection of web service providers driven by composite QoS. In *Proc. of 2011 IEEE Symposium on Computers and Communications (ISCC)*, pages 55–60. IEEE.

Strunk, A. (2010). Qos-aware service composition: A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74.

Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., and Zhang, J. (2007). Autonomous service level agreement negotiation for service composition provision. *Future Gener. Comput. Syst.*, 23(6):748–759.

Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Trans. Web*, 1(1).

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Trans. on Software Engineering*, 30(5):311–327.